Pour obtenir ces modèles mathématiques, j'ai utilisé **la régression linéaire**, qui est une technique permettant de trouver la relation entre une variable indépendante (**Col1**) et une variable dépendante (**Col2, Col3, Col4**). Voici les étapes détaillées :

**1. Préparer les données**

Nous avons un jeu de données avec une colonne **Col1** (variable indépendante) et trois colonnes **Col2, Col3, Col4** (variables dépendantes).

**🔹 Format des données**

| **Date** | **Col1** | **Col2** | **Col3** | **Col4** |
| --- | --- | --- | --- | --- |
| 2025-03-03 | 31.66 | 7.803 | 7.753 | 5.179 |
| 2025-03-04 | 31.98 | 7.827 | 7.777 | 5.195 |
| 2025-03-05 | 32.62 | 7.851 | 7.801 | 5.211 |
| ... | ... | ... | ... | ... |

Nous allons utiliser **Col1** pour estimer **Col2, Col3 et Col4** séparément.

**2. Utiliser la régression linéaire**

La régression linéaire cherche une équation de la forme :

Y=aX+bY = aX + bY=aX+b

où :

* **XXX** est la variable indépendante (**Col1**)
* **YYY** est la variable dépendante (**Col2, Col3 ou Col4**)
* **aaa** est le coefficient de régression (pente)
* **bbb** est l'interception (valeur de Y quand X = 0)

Nous allons entraîner trois modèles de régression, un pour chaque colonne cible (**Col2, Col3 et Col4**).

**3. Entraîner le modèle**

Nous utilisons la **régression linéaire simple** avec scikit-learn. Voici le code utilisé :

python

CopierModifier

from sklearn.linear\_model import LinearRegression

import numpy as np

X = df["Col1"].values.reshape(-1, 1) # Variable indépendante

models = {}

equations = {}

for col in ["Col2", "Col3", "Col4"]:

y = df[col].values # Variable dépendante

model = LinearRegression()

model.fit(X, y) # Entraînement du modèle

models[col] = model

# Récupérer la pente (coef\_) et l'interception (intercept\_)

a = model.coef\_[0]

b = model.intercept\_

equations[col] = f"{col} = {a:.6f} \* Col1 + {b:.6f}"

**4. Interprétation des résultats**

Après l'entraînement, nous avons obtenu les modèles :

* **Col2 = 0.149927 × Col1 + 3.043445**
* **Col3 = 0.149927 × Col1 + 2.993445**
* **Col4 = 0.100028 × Col1 + 2.003465**

Cela signifie que chaque colonne **Col2, Col3 et Col4** a une relation linéaire avec **Col1**. Par exemple :

* Si **Col1 augmente de 1 unité**, alors **Col2 et Col3 augmentent d'environ 0.1499**, et **Col4 augmente d'environ 0.1000**.

**5. Vérifier la qualité du modèle**

On peut calculer le **coefficient de détermination R2R^2R2** pour voir à quel point le modèle est bon :

python

CopierModifier

for col in ["Col2", "Col3", "Col4"]:

r2\_score = models[col].score(X, df[col])

print(f"R² pour {col} = {r2\_score:.4f}")

Un **R2R^2R2** proche de **1** signifie que le modèle est bien ajusté.

**6. Utiliser les équations pour faire des prédictions**

Si on a une nouvelle valeur pour **Col1**, par exemple **Col1 = 32.5**, on peut prédire :

* **Col2 = 0.149927 × 32.5 + 3.043445 ≈ 7.91**
* **Col3 = 0.149927 × 32.5 + 2.993445 ≈ 7.86**
* **Col4 = 0.100028 × 32.5 + 2.003465 ≈ 5.25**